COMAL-80 Quick Reference (draft)

This concise quick reference groups COMAL commands by library. Syntax, a one-line description, and a tiny example are provided.

Language (core COMAL syntax & usage)

- **Variable names & types:** Must begin with a letter; may include letters, digits and special characters. Common suffixes: `$` = string variable (e.g. `name$`), `#` = integer numeric (e.g. `i#`). Names like `a`, `a#`, `a$`, and `a()` refer to the same base name with different forms.
- **Assignment:** Use `:=` to assign expressions to variables. (Typing `=` will be converted by COMAL.) Example: `average := (a + b) / 2`.
- **Strings & concatenation:** Strings in quotes. Concatenate with `+`. Example: `full$ := name$ + " " + city$`.
- **Arrays / DIM:** Numeric arrays: `DIM a(50)` (or `DIM a(3,4)` for multi-dimensional). String storage: `DIM name$ OF 20` (space for 20 chars). Indexing uses parentheses: `a(1)`, `text$(i)`.
- **Comments:** Prefix with `//` for a comment line.
- **Multiple statements on one line:** Separate statements with a semicolon `;`. Example: `forward(50); left(90)`.
- **PRINT / INPUT quick rules:** `INPUT "prompt": varlist` reads values. `PRINT` accepts separators: comma `,` (column spacing per `ZONE`), semicolon `;` (prints with no extra line, adds space as separator), or no separator (concatenates). Example: `PRINT "Sum =", a+b`.
- **Control flow:**
  - Conditional: `IF <cond> THEN` … `ELIF` … `ELSE` … `ENDIF`. Single-line: `IF <cond> THEN <statement>`.
  - CASE: `CASE <expr> OF ... WHEN ... OTHERWISE ... ENDCASE`.
  - Loops: `REPEAT` … `UNTIL <cond>`; `WHILE <cond> DO` … `ENDWHILE` (or single-line `WHILE <cond> DO <statement>`); `FOR var := start TO end [STEP step] DO` … `ENDFOR` (single-line `FOR ... DO <statement>` also supported).
  - Structured loop: `LOOP` / `EXIT WHEN` / `ENDLOOP`.
  - Jump: `GOTO label` with labeled lines.
- **Procedures & Functions:**
  - Procedure: `PROC name(params)` … `ENDPROC name`. Use `REF` for pass-by-reference parameters and `CLOSED` to declare a closed procedure. Call: `name(arguments)` or use as statement.
  - Function: `FUNC name(params)` … `RETURN <expr>` … `ENDFUNC name`. Use like: `x := name(a#,b#)`.
- **Error handling:** `TRAP` / `HANDLER` / `ENDTRAP` with `ERR`, `ERRFILE`, `ERRTEXTS` available.
- **Useful system orders / misc:** `DIM`, `PEEK`, `POKE`, `SYS` (machine call), `NULL`, `STOP`, `END`, `TRACE`.
- **Operators:** Arithmetic `+ - * /`, integer ops `DIV`, `MOD`. Logical: `NOT`, `AND`, `AND THEN`, `OR`, `OR ELSE`, membership `IN`. Bit ops: `BITAND`, `BITOR`, `BITXOR`. Comparison operators (<, >, <=, >=, =, <>) are used in conditions.
- **Common built-in functions:** numeric & string helpers: `INT`, `SGN`, `SQR`, `PI`, `COS`, `SIN`, `TAN`, `ATN`, `CHR$`, `STR$`, `ORD`, `VAL`, `LEN`, `TIME`, `RND`/`RANDOMIZE`.
- **Quick examples:**
  - `// comment` (comment)
  - `INPUT "Name:": name$`
  - `FOR i# := 1 TO 10 DO PRINT i#; ENDFOR`
  - `PROC swap(REF a#, REF b#) ... ENDPROC swap`

This section summarizes the core language features you always have available (no `USE` required).

Commands Used Before and During Program Entry

USE: (core — no package required)

- **NEW:** Clears program and working memory; resets system vars. Example: `NEW`
- **AUTO:** Enables automatic line numbering; formats: `AUTO`, `AUTO start`, `AUTO start,step`. Example: `AUTO 100,2` (100,102,…)
- **RENUM:** Renumber program lines. Formats: `RENUM`, `RENUM start`, `RENUM from,to,step`. Example: `RENUM 2000,5`

Program Editing Commands

USE: (core — no package required)

- **EDIT:** Edit lines; forms: `EDIT`, `EDIT n`, `EDIT n-m`, `EDIT procname`. Example: `EDIT 210-290`
- **FIND:** Search program for text or name; can limit by line range or procedure. Example: `FIND "John"` or `FIND 200-500 "John"`
- **CHANGE:** Search-and-replace text in program (interactive options: accept/change/skip). Example: `CHANGE 50-200 "xi","xstart"`
- **DEL:** Delete program lines or named procedures. Example: `DEL 20` or `DEL 40,200-280`
- **SCAN:** Prepass program structure check; reports syntax/structure errors. Example: `SCAN`

Other Useful Commands

USE: (core — no package required)

- **SETEXEC:** Controls printing of `EXEC` before procedure calls. `SETEXEC-` hides, `SETEXEC+` shows. Example: `SETEXEC+`

Memory & Disk Utilities

USE: (core — no package required)

- **SIZE:** Show bytes used: `prog data fre`. Example: `SIZE`
- **CAT:** Catalogue diskette contents; accepts pattern and device. Example: `CAT "t*"`
- **DIR:** Like `CAT` but can be used as statement in programs. Example: `DIR`

I/O and Program Transfer

USE: (core — no package required)

- **LIST:** Print or save program/procedure. `LIST`, `LIST n-m`, `LIST name`, `LIST "file"` (to disk). Example: `LIST "myprog"`
- **ENTER:** Load a program saved via `LIST` (erases current memory). Example: `ENTER "lst.name"`
- **MERGE:** Merge a saved program segment into current memory (non-destructive). Example: `MERGE "segname"`

Quick Index (Alphabetical)

- **adsr:** (sound) Set ADSR envelope for a voice. USE: `USE sound`
- **AUTO:** (core) Enable automatic line numbering. USE: (core)

- **bell:** (system) Produce audible bell. USE: `USE system`
- **back / bk:** (turtle) Move turtle backward. USE: `USE turtle`
- **cat / dir:** (core) Catalogue directory. USE: (core)
- **change:** (edit) Search-and-replace in program. USE: (core)
- **circle:** (graphics) Draw a circle. USE: `USE graphics`
- **clearscreen / cs:** (turtle/graphics) Clear graphics screen. USE: `USE turtle` or `USE graphics`
- **close:** (file) Close file stream. USE: (core)
- **datacollision:** (sprites) Test sprite/graphics collision. USE: `USE sprites`
- **define:** (sprites) Define shape/drawing. USE: `USE sprites`
- **discard:** (machine-code) Remove linked machine-code packages. USE: (machine-code)
- **draw / drawto:** (graphics) Draw relative/absolute lines. USE: `USE graphics`
- **edit:** (edit) Edit program lines or procedures. USE: (core)
- **enter:** (file) Load saved program (erases memory). USE: (core)
- **find:** (edit) Search program text. USE: (core)
- **getscreen / setscreen:** (system) Save/restore text screen. USE: `USE system`
- **gate:** (sound) Start/stop sustain on a voice. USE: `USE sound`
- **hardcopy:** (system) Print text screen to printer. USE: `USE system`
- **hideturtle / ht:** (turtle) Hide turtle cursor. USE: `USE turtle`
- **identify:** (sprites) Attach drawing to sprite number. USE: `USE sprites`
- **inkey$:** (system) Non-blocking keyboard read. USE: `USE system`
- **joystick:** (control) Read joystick direction/button. USE: `USE joysticks`
- **keywords_in_upper_case:** (system) Toggle keyword case. USE: `USE system`
- **LINK:** (machine-code) Load machine-code object. USE: (machine-code)
- **loadfont / savefont:** (font) Load/save user font. USE: `USE font`
- **list:** (core) Print or save program. USE: (core)
- **moveto / move:** (graphics) Move drawing cursor. USE: `USE graphics`
- **movesprite:** (sprites) Move sprite with steps/collision. USE: `USE sprites`
- **names_in_upper_case:** (system) Toggle name case. USE: `USE system`
- **note:** (sound) Play musical note string. USE: `USE sound`
- **open file:** (file) Open file stream (WRITE/READ/RANDOM). USE: (core)
- **paint:** (graphics) Flood-fill area. USE: `USE graphics`
- **pencolor / pc:** (turtle) Set drawing color. USE: `USE turtle`
- **penup / pu / pendown / pd:** (turtle) Stop/start drawing. USE: `USE turtle`
- **playscore / waitscore:** (sound) Play composed score and wait. USE: `USE sound`
- **plot / plottext:** (graphics) Plot point or text in graphics coords. USE: `USE graphics`
- **print file / input file:** (file) ASCII I/O on opened files. USE: (core)
- **putcharacter / getcharacter:** (font) Read/write character raster. USE: `USE font`
- **renum:** (core) Renumber program lines. USE: (core)
- **setrecorddelay:** (system) Delay for disk writes. USE: `USE system`
- **setprinter:** (system) Configure printer/serial attributes. USE: `USE system`
- **setexec:** (core) Control EXEC display. USE: (core)
- **serial:** (system) Route output to serial device. USE: `USE system`
- **show sprite / showsprite:** (sprites) Make sprite visible. USE: `USE sprites`
- **size:** (core) Show memory usage. USE: (core)
- **soundtype:** (sound) Set waveform for voice. USE: `USE sound`
- **save / save shape:** (core / sprites) Save program or sprite shape. USE: (core) / `USE sprites`
- **setheading / seth:** (turtle) Set absolute heading. USE: `USE turtle`
- **spritecolor:** (sprites) Set sprite color. USE: `USE sprites`
- **write file / read file:** (file) Binary record I/O. USE: (core)

Turtle Graphics (USE turtle)

USE: `USE turtle`

- **back(L) / bk(L):** Move the turtle L units backward. Example: `back 50`
- **forward(L) / fd(L):** Move the turtle L units forward. Example: `forward 75`
- **penup / pu:** Stop drawing while moving. Example: `penup`
- **pendown / pd:** Resume drawing while moving. Example: `pendown`
- **left(D) / lt(D):** Turn turtle left D degrees. Example: `left 90`
- **right(D) / rt(D):** Turn turtle right D degrees. Example: `right 45`
- **setheading(H) / seth(H):** Point turtle to absolute heading H (0 up, 90 right). Example: `setheading 0`
- **home:** Move turtle to screen center. Example: `home`
- **hideturtle / ht:** Hide the turtle cursor. Example: `hideturtle`
- **showturtle / st:** Show the turtle cursor. Example: `showturtle`
- **pencolor(C) / pc(C):** Set drawing color to C (see Appendix B color codes). Example: `pencolor(3)`
- **clearscreen / cs:** Clear the graphics screen. Example: `clearscreen`

Graphics Package (USE graphics)

USE: `USE graphics`

- **graphicscreen(n):** Select graphics screen/mode (e.g., 0/1). Example: `graphicscreen(1)`
- **moveto(x,y):** Move drawing cursor to absolute coords (no draw). Example: `moveto(10,20)`
- **drawto(x,y):** Draw line from current cursor to absolute (x,y). Example: `drawto(100,50)`
- **move(dx,dy):** Move cursor by relative offset (no draw). Example: `move(10,0)`
- **draw(dx,dy):** Draw line by relative offset. Example: `draw(0,50)`
- **circle(xc,yc,r):** Draw circle centered at (xc,yc) radius r. Example: `circle(160,100,70)`
- **paint(x,y):** Fill enclosed area containing point (x,y). Example: `paint(160,100)`
- **plottext(x,y,text):** Draw text at graphics coords. Example: `plottext(0,100,"COMAL graphics")`
- **plot(x,y):** Plot a single point at (x,y). Example: `plot(50,50)`
- **viewport(…) / window(…):** Define drawing coordinate system. Example: `viewport(0,159,0,99)`

Sprites Package (USE sprites)

USE: `USE sprites`

- **define(drawingno, string$):** Define a drawing (from DATA or string). Example: `define(1,drawing$)`
- **identify(spriteno,drawingno):** Attach drawing to sprite number. Example: `identify(1,1)`
- **spritecolor(spriteno,color):** Set sprite color. Example: `spritecolor(1,2)`
- **spritepos(spriteno,x,y):** Place sprite at (x,y). Example: `spritepos(1,50,100)`
- **showsprite(spriteno):** Make sprite visible. Example: `showsprite(1)`
- **movesprite(spriteno,x,y,steps,collisionFlag):** Move sprite to (x,y) in N steps; last arg controls collision behavior. Example: `movesprite(1,250,150,200,0)`
- **spritesize(spriteno,doubleH,doubleW):** Enlarge sprite vertically/horizontally. Example: `spritesize(1,TRUE,TRUE)`
- **saveshape(drawingno,filename) / loadshape(drawingno,filename):** Save/load drawing to disk. Example: `saveshape(1,"sp0.rabbit")`
- **datacollision(spriteno,TRUE):** Test collision with graphics; used in loops. Example: `WHILE NOT datacollision(1,TRUE) DO NULL`

Sound Package (USE sound)

USE: `USE sound`

- **soundtype(voice,type):** Set waveform type for voice (0-4). Example: `soundtype(1,3)`
- **volume(v):** Set global/output volume (0-15). Example: `volume(10)`
- **adsr(voice,a,d,s,r):** Set ADSR envelope params (0-15). Example: `adsr(1,2,4,8,3)`
- **note(voice,code$):** Prepare and play a note string (e.g., “c4”, “z” for pause). Example: `note(1,"c4")`
- **gate(voice,0|1):** Start/stop sustain phase (1=start,0=stop). Example: `gate(1,1)`
- **playscore / waitscore:** Play composed score and wait for completion. Example: `playscore` then `WAIT WHILE NOT waitscore()`
- **frequency(code$):** Convert note code to frequency (used in examples). Example: `f:=frequency("c4")`

File Handling (core)

USE: (core — no package required)

- **OPEN FILE #n,“name”,MODE:** Open file stream `n` with `WRITE`, `READ`, `APPEND`, or `RANDOM size`. Example: `OPEN FILE 3,"datafile",WRITE`
- **CLOSE FILE n:** Close stream `n`. Example: `CLOSE FILE 3`
- **PRINT FILE n: vars** / **INPUT FILE n: vars** — ASCII I/O to/from opened sequential/random files. Example: `PRINT FILE 2: items` / `INPUT FILE 2: items`
- **WRITE FILE n: vars** / **READ FILE n: vars** — Binary (compact) I/O (faster, smaller). Example: `WRITE FILE 4,12: nostexts` / `READ FILE 4,12: nostexts`
- **Notes:** Use `RANDOM` when fixed-length records are needed (e.g., `OPEN FILE 5,"text",RANDOM 42`). Prefer `WRITE FILE` for disk storage.

Control Ports: Paddles & Joysticks (USE paddles / USE joysticks)

USE: `USE paddles` / `USE joysticks`

- **paddle(port,a_paddle,b_paddle,a_button,b_button):** Read paddle pair on `port` (1 or 2). Returns 0–255 positions and button flags (0/1). Example: `paddle(2,a,b,abtn,bbtn)` then `PRINT a,b,abtn,bbtn`
- **joystick(port,direction,button):** Read joystick on `port` (1 or 2). `direction` is a numeric code (0..8) mapping compass points; `button` is 1 when pressed. Example: `joystick(2,dir,btn)` then `IF dir>0 THEN setheading((dir-1)*45)`

Light Pen (USE lightpen)

USE: `USE lightpen`

- **readpen(x,y,ok):** Read a light-pen event; returns coordinates and `ok` flag. Typical pattern: `REPEAT readpen(x,y,ok) UNTIL ok` then use `(x,y)`.
- **delay(n) / timeon(n):** Tuning parameters controlling sample/hold and sensitivity; set small values for fast drawing. Example: `delay(1) timeon(1)`
- **Calibration:** calibrate by reading pen at known screen corner and storing offsets (examples in text show calibrating then using `moveto(x,y)` or `drawto(x,y)`).

System Package (USE system)

USE: `USE system`

- **textcolors(border,background,text):** Set text-screen color scheme. Example: `textcolors(0,2,1)`

- **keywords_in_upper_case(TRUE|FALSE):** Toggle keyword display case. Example: `keywords_in_upper_case(FALSE)`
- **names_in_upper_case(TRUE|FALSE):** Toggle name-case. Example: `names_in_upper_case(TRUE)`
- **inkey$():** Read keyboard input (returns string). Example: `k$:=inkey$()`
- **gettimes():** Return current clock string hh:mm:ss.t. Example: `PRINT gettimes()`
- **getscreen(s¿/*setscreen*¿¿):** Save/restore text-screen image to/from string (use `DIM s$ OF 1505`). Example: `getscreen(screen$)`
- **hardcopy(unit$):** Print the text screen to `unit$(printer). Example:hardcopy("lps")`
- **currow / curcol:** Functions returning current cursor row/column. Example: `r:=currow()`
- **bell(duration):** Produce bell sound for `duration` (1..255). Example: `bell(10)`
- **setprinter(attributes$):** Configure printer/serial routing and device attributes. Attribute string controls device, formfeed, linefeed, etc. Example: `setprinter("u5,c1")`
- **serial(TRUE|FALSE):** Route output to serial device (`TRUE`) or normal printer (`FALSE`). Example: `serial(TRUE)`
- **setrecorddelay(ms):** Pause COMAL for `ms` milliseconds on writes to random files to allow DOS to finish disk block writes. Default: `setrecorddelay(50)` unless IEEE module present (then `0`). Example: `setrecorddelay(50)`

Font Package (USE font)

USE: `USE font`

- **linkfont:** Reserve user font area and enable new double character sets (0/1). Must be run before `putcharacter` or `loadfont` when creating a user font.
- **loadfont(filename¿/*savefont*¿¿):** Load or save a double character set from/to disk. Example: `loadfont("myfont")`
- **getcharacter(set#,char#,raster$) / putcharacter(set#,char#,raster$):** Read or write an 8x8 raster for a character. Example: `putcharacter(1,28,rasters$)`
- **Notes:** Use `linkfont` (or `loadfont`) first; `discard` or `NEW` removes a user font.

Machine-Code Packages & Drivers

USE: (machine-code packages are attached with `LINK` then enabled with `USE <name>`)

- **LINK** "**filename**": Load a machine-code/object package from disk into working memory. Example: `LINK "obj.driver"`
- **USE packageName:** Make the linked package's name available to COMAL (activates its procedures). Example: `USE driver`
- **DISCARD:** Remove all linked machine-code packages from working memory (COMAL program remains). Example: `DISCARD`
- **Notes:** A linked machine-code package can be saved together with the COMAL program via `SAVE` and restored with `LOAD`.

Language / Localization Packages

USE: `USE english` / `USE dansk`

- **USE english / USE dansk:** Switch built-in COMAL messages and error text to English or Danish. Example: `USE english`
- **Notes:** These packages contain no procedures — they only set localized error/messages.